

SRM SCALABILITY/PERFORMANCE REVIEW

CD-doc-3163-v0

4/3/2009

Gene Oleynik, Matt Crawford, Gerd Behrmann, Catalin Dumitrescu,
Suzanne Gysin, Tanya Levshina, Pedro Salgado, Timur Perelmutov

A review of the SRM front-end to dCache was performed on March 13th and March 18th 2009 with the goal of identifying and recommending directions to improve the performance and scalability of the SRM. The charge to the reviewers was:

I am writing to invite you to participate in a software review of the Storage Resource Manager (SRM) implementation that is a part of the dCache storage system. This SRM is a crucial component of the LHC software suite; it is used to transfer files between storage systems and to manage space on those systems. Performance at high transaction rates is a concern as LHC needs continue to increase. Currently US-CMS T1 has measured a rate of about 5 transfers/s. The experiments expect that increased transaction rates will be required as the LHC data-taking and analysis ramps up during the first few years of the LHC. The SRM implementation will have to meet those scaling requirements.

This review will focus on the performance and scalability of the SRM system through examination of its architecture and design, algorithms, and maintainability. The charge to the reviewers is to analyze the current architecture and plans; identify areas that may limit the transaction rate now or in the foreseeable future; and to make recommendations for improvements. The briefing material will contain some proposed performance and scalability improvements. In addition to reviewing these suggestions for correctness and merit, the reviewers are asked to identify bottlenecks and suggest additional performance and scalability improvements.

Material will be provided in advance of the review. We will have two half-day meetings in March. The first will be a presentation to the reviewers to familiarize them with SRM, to answer their questions and to provide any other material the reviewers require. The second half-day will be for the reviewers to question the developers and for open discussion driven by the reviewers. The last two hours of this day will be used to collect and write up recommendations. Additional findings and recommendations can be submitted after the meeting.

The reviewers and their affiliations were:

Catalin Dumitrescu	US-CMS T1, Fermilab
Gerd Behrmann	dCache, NGDF T1
Pedro Salgado	BNL T1
Tanya Levshina	OSG, Fermilab
Suzanne Gysin	Computing Enabling Technologies, Fermilab

The software author was Timur Perelmotov and the review organizers were Matt Crawford and Gene Oleynik.

Material from the review can be located through the web page:

<http://home.fnal.gov/~timur/srm/review/>

While the review was being organized, it became known to us that the SRM transfer requirement for CMS has risen from 2Hz to 100 Hz. The recommendations have taken this into consideration.

The conclusions and recommendations from the review panel are given in the next section. Detailed minutes from the meeting are given in the last section.

Many thanks are due to all of the reviewers for their interest and time spent in providing excellent and thoughtful recommendations and to Timur Perelmotov for providing the review material.

CONCLUSIONS AND RECOMMENDATIONS

The conclusions and recommendations fall under two categories: performance and performance scalability, and maintainability and other concerns. It is recognized that the maintainability will impact performance and the ability to react to changing requirements in an adequate time.

PERFORMANCE

The general feeling of the reviewers was that the current source of the bottleneck of 5 Hz transfers in SRM transfers has not been positively identified, and indeed might not even be in SRM. There is evidence that the SRM is not CPU bound when running flat out (from both NGDF and BNL. See notes). Aside from handling authentication, SRM should use little CPU per request – it mainly orchestrates actions elsewhere in dCache.

With this in mind, the reviewers had the following comments and recommendations:

1. The 100 Hz requirement needs to be clarified. What exactly does it mean (under what loads, etc.)?
2. The current transfer throughput should be baselined. There is some anecdotal information that increasing the backlog depth for the request listen() socket may have increased the performance from 5 to 20 Hz on the USCMS-T1 system.
3. The plan to load balance SRM horizontally should be pursued at a high priority. The consensus of the reviewers is that this change alone will not achieve desired rate, but that it is a necessary piece of a scalable solution. It would, in addition, provide a level of redundancy in SRM/dCache. The reviewers would like to see a more detailed design for scaling horizontally. A prototype in the time frame of month and some real transaction processing would be in order.
4. SRM (and dCache) should evaluate using Terracotta in front of distributed

database accesses. This should be done prior to a prototype and a detailed horizontal scaling design as it will affect the prototype/design.

5. Identify and use proper tools to profile what is going on, identify bottlenecks is very important, especially for when things go online.
6. Get more hard data on where the bottlenecks really are. At what rate can you push SRM ping until the CPU is maxed out? How far can the name service be pushed?
7. There should be a joint effort to identify key tuning parameters and performance indicators such that SRM/dCache performance can be monitored and bottlenecks can be identified not only in test, but also in production systems.
 - a. The SRM/dCache developers should make a complete list of tuning parameters (including, for example: the various thread pools, what they do, impact of their sizes; request listen backlog depth) and their effect on performance under specific circumstances.
 - b. A joint effort should be made to develop monitoring for key performance indicators (CMS, OSG and BNL are interested in doing this).
8. Increase staff - a person with dedicated time allocated for performance analysis and tuning. Look at root causes, compare to CERN.
9. Work with clients (FTS and others) to use exponential backoff on retries
10. Pursue improving the performance of the GSI authentication (of the CPU used, 90% of it is in authentication related activity). Follow up on request for CDIGS to port gt4 authentication/delegation speedups to jglobus-CoG (Catalin notes jglobus in hibernation).
11. Pursue using SSL as a more efficient means of authentication than GSI. CERN has approached dCache about this. SRM should talk with CERN and others to start discussions of replacing GSI authentication with SSL.
12. Decouple the dependency of SRM on other components as much as possible
 - a. Reduce the dependency on other components (e.g. reduce queries to other components via caching)
 - b. Prevent SRM from overwhelming dCache components (e.g. by limiting the number of outstanding requests to those components. For example the number of outstanding srmls pnfs operations is limited by the number of srmls threads)
13. Consider separating SRM by classes of service activity (get, put, etc). In the future consider protecting each class of activity from being undercut by others.
14. Srmls has a bottleneck as it must go through pnfs. One suggestion is for the dCache team to consider adding an ls function to the pnfs manager.
15. It should be possible for site administrators to manage or to isolate VOs, roles, or users that are impacting performance.
16. Timeout handling should be reviewed in order to reduce the impact of requests that have timed out but continue to use resources.

MAINTAINABILITY AND OTHER RECOMMENDATIONS

1. The dCache collaboration should come up with a way to quickly turn-around SRM/dCache releases or patches (The reviewers would like to see decouple SRM/dCache releases and rpms, but the two are tightly coupled by necessity – see discussion in notes). This will be critical as the LHC comes online.
2. Exception handling is not very good as logged - site administrators are having a difficult time analyzing problems. The same is true with messages SRM returns back to the client. e.g. timeout when there is a missing host cert. All should look at SRM client and exit codes (fails with 0 exit code so can't script it but must parse output). Be considerate of backwards compatibility,
3. Provide unit regression tests for basic functionality (and potentially profiling) included in code repository for validating releases. A practical necessity is refactoring (possibly by activities). BNL has interest in contributing to this.
4. Perform code walk-throughs after horizontal scaling by small pieces: request handler, job persistency, scheduler, authentication, database performance
5. Reduce the barrier to learning dCache/SRM code base:
 - a. Make a pass through the code to make it more readable in a standard format, code structure, and naming convention. Parts of the code seem to have been copy and paste (excluding generated code). Provide javadoc on all code and UML diagrams. BNL is interested in contributing in this area.
 - b. Using J2EE patterns and open source packages would help increase the readability and maintainability of the code.
 - c. In addition to existing sequence diagrams, add flow maps that show the components invoked by the requests.
6. The system should tune as many parameters as possible dynamically based on the configuration (system and hardware capabilities).
7. Careful attention should be paid in maintaining compatibility of the SRM client with all SRM compliant servers.

MANAGEMENT

1. A plan with timelines and resource commitments should be established and agreed to by all involved parties
2. Plan should be followed up and tracked by Management

NOTES FROM THE REVIEW MEETINGS

SRM/dcache review 2009-03-13

Conferencing conquered at 8:15

Gerd Behrmann-remote CPH

Catalin Dumtresco, Suzanne Gysin, Pedro Salgado, Gene Oleynik, Timur Perelmutov, Matt Crawford.

Gene shows proposed agenda.

Timur: Raised the backlog count in the listen() syscall from 10 to 10,000 and found that they could handle at least 3,000 simultaneous clients using about 50% of the CPU.

jglobus-CoG: room for improvement in GSI overheads.

Matt: Requested CDIGS to port gt4 authn/delegation speedups to jglobus-CoG.

Catalin: jglobus is in hibernation.

ASN.1 parser consumes a lot of time. New Bouncy Castle lib is faster there.

Gerd: Aside from the authentication, SRM uses little CPU per request. Mainly it orchestrates actions elsewhere in dCache.

Pedro: Memory, CPU not overloaded. It's synchronization. 4-core machine 8GB CPU stable at 40% ... but load jumped to 12. He would not care to have remote profiling conducted from here, but he could do two things: Profile it on his site; Check logs after next period of overload and see what sorts of requests are numerous.

Suzanne: Best investment to make now is profiling tools. Whatever we observe now in our systems is likely to change when experiments really start up.

Gerd: There were loads of 600-700 at FZK due to thread contention. This was a direct result of a large thread pool. Decreasing it helped. Asks BNL to go to 1.9 ... which they did 2 weeks ago, to 1.9.x (x unk.)

Pedro/Suzanne/Timur: Do GSI in Apache (with C lib), serialize credential, use "filter," pass to Tomcat.

T/P: Splitting SRM across machines would not lead to inter-machine synchronization problems. Database is seldom an obstacle.

Timur: srmls is the worst. There's no dCache function to "ls," have to go through pnfs.

Suzanne: cache results? Gerd: Don't give up on having the ls function added to pnfsmanager. Srmls is synchronous in srm - one continuous tomcat operation. Clients expect ls to be a fast operation and may disconnect. Tomcat will not notice the disconnect until it tries to send the results. In the newest versions, srmls is asynchronous - committed to trunk 2 days ago.

Timur: New state machine for ls: When work is done, it goes to READY-QUEUED state. When the client next polls for status, it gets the results and request goes directly to DONE. The old dCache-SRM client doesn't know what to do with a QUEUED reply from srmls, so that update will have to go out sooner. (ETA: 2 weeks) Gerd: Can we handle the easy requests synchronously and avoid a second authn? Timur: Not now.

==== Top concerns of reviewers ====

[second-pass discussion]

Pedro: After we have real LHC data and more user activity, he would like to have a way of protecting the services, in order to be able to make service guarantees to different classes of users - each class gets a service floor that won't be undercut by high loads from other classes. He needs a DoS protection. With parallel SRM servers, would this be easier or harder?

Timur replies: DNS-based load balancing won't even tie a given client's polling to one server. Separately-named SRM endpoints can tie different classes to different servers. There is a rudimentary notion of priority of different requests (based on user credential).

The features of this could be enriched ...

Suzanne: What tools exist to track and shut down user identities that are grossly overloading the system? Timur, Pedro: a few.

Pedro: maintainability. Coupling of SRM component with release schedule of the rest of dCache.

Pedro: Barrier to learning the code is high because it's all tangle together in SVN. S: How many people submitting code? Just to SRM, 2 (TP & DL). P: Use of J2EE patterns and open source packages would be a virtue. T: There is now a separation between the dCache-dependent parts of SRM and the generic parts. T: Yes, they could improve modularity and use standard components more. [G: Yes, there's movement to this sort of cleanup, but it loses out to bugfixing. S: Spring? G: yes, starting to use Spring.]

Suzanne: Development is boxed in by the spec. and by GSI.

S: When the LHC comes on-line with a lot of data, there will be "a whole slew of different problems." There will have to be a fast release process for patched versions. Good bug reporting & tracking tools will be essential. LHC MTBF may be low and if data distribution fails while beam is present, there will be massive widespread unhappiness. [G: Maintainability, yeah!]

G: Views SRM as an integrated part of dCache and is hesitant about a decoupled release program. Frequency of dCache releases needs to be addressed even without reference to SRM. Problems elsewhere in dCache can be just as significant in their impact. Must try to get DESY into a mode in which bugfix releases are as fast as possible, and as automated as possible. And by the way, Gerd sometimes deploys patches on NDGF first, then submits them for review. P: (Expression of alarm)

Continues to advocate decouple release schedule for SRM. T: dcache already allows (up to a point) for different components to run different releases - pool nodes run X, head nodes X', SRM servers X".

S: For patches you don't need a whole RPM, just a WAR file.

G: Understand, please, that dCache has its own application environment (cells) alongside the Tomcat container. SRM is a bit different from all the other components of dCache. T: Tried several years ago to package Tomcat with dCache. But then there were two classloaders, and things got rather difficult

[P: Can you do a new release in an hour, if you have the fix? T: No way. We have to synchronize with DESY, and so on. P/S: You may need to!]

Catalin: We want SRM clients to be compatible with all implementations. T: Recent incompatibility incident stemmed from a CERN change to gsiftp which had some deviation regarding checksum verification.

C: We recently identified several parameters that affected performance. Could we get all of them defined and documented? After learning them, he was able to keep the system running much more stably at higher loads. Perhaps the defaults could be chosen at install time as functions of the server hardware or other factors. [Provide advice to the user as a file to read with the eyes, or as a software calculation at installation time?]

C: Pinning. We have to make sure this works correctly. And transfers should not fail solely due to pinning problems. [Tangentially related to scaling.]

C: (in resp. to Gene Q) USCMS does not want different SRM endpoints at different DNS names. Multiple (3-10) servers behind a load balancing mechanism is acceptable. [Pedro wants to also have multiple names providing separate service instances.]

Matt: If Jon's "20-100 Hz" means file transfers per second, the additional load from clients polling for status, doing srmls, and other functions, is an undefined and unaccounted-for variable. [When negotiating our target, identify the conditions of measurement. Make the goal very realistic and reality-based.]

C: Would like SRM performance less sensitive to loads in other parts of dCache, especially pnfs. T: pnfs is so central we can't do much without it. Two areas of progress, though: Chimera (DESY) and Berkeley db-based pnfs (Vladimir). [Doing ls through pnfs involves a lot of round-trips. Providing a service in dCache eliminates many of them. Tigran has pointed out that caching could be turned on in a system that doesn't modify pnfs, and that the SRM server is such a system. There may be some race conditions possible that need to be checked first.]

<br duration="10m" />

Gerd: Agree w/ Pedro about "performance isolation." (Between VOs, roles, or even users.)

G: Priorities. Can't communicate them to the rest of dCache. One SRM thread could flood the rest of dCache. [Then the only way to make sure your SRM priorities are respected is to make sure SRM's total of current + waiting operations for dCache is bounded below dCache's capacity.]

G: Have seen problems with the thread pool sizes. Asks T. whether the asynchronous jobs are fully asynchronous, or only mostly? If the former, the thread pool can be $O(\# \text{ cores})$. [T: GET, PUT, COPY 100%. Database locks when, say, storing a credential. G: In that case, smaller thread pools would be fine. T: For GET & PUT, yes. An important factor is the number of READY state requests(?). For COPY, there's a "Running without thread" state to keep a limit on the number of parallel copies being run.]

[P: Database inserts & updates should be fast ... unless the rows are large and the DB has to allocate pages. T: Using database access threads to limit the number of db operations outstanding. M: But then threads will block longer sometimes, waiting for a db thread. S: Make more such threads?

T: After fixing some "stupid schema tricks" db performance got a lot better.]

G: Handling of timeouts - working on a request after the client has gone away. Can put a lot of load on the back-end system (pnfs).

G: To what extent can we limit the number of concurrent jobs we prepare at once? Looks like SRM will try all available jobs at once.

G: Modularity & code re-use. Some of the things that ought to be used in dCache/SRM became available since the work started. Move toward using them.

G: (Maybe not particular to this review.) SRM sometimes has to artificially work around features missing from the core of dCache. Space management is an example. This is a result of the historical path of development.

T: Integrate pool selection, space mgmt, pinning in one service in dCache. This is could be a feature for Cache 2.0.

G: More hard data about where the bottlenecks are! This will help us know whether more cores will help. T: Service fails before it can use all the CPU resources available. This was especially so when the listen() backlog was small.

T: Setting a number of threads for srmls scheduling sets the number of pnfs operations in progress at a time. G: Would like to have similar regulation of other operations.

=====

Discussion of site availability tests. Tests do not accept long queuing - limit is five minutes. T: This is unfair. The system SHOULD queue requests if load is high! P: Dashboard at BNL. C: Pages due to SRM failures are reduced. P: At BNL also. When SRM timeouts are seen, it's always a dCache problem. Want to have probes on dCache components to detect and isolate these quickly. T: Propagation of errors is poor. In Enstore terms, a NOACCESS tape leads to SRM failures, but they are unexplained failures. Helpdesk ticket always points to SRM at first.

P: More and more agents examining the system state and taking action in response.

GO/C: The 100Hz figure is the T0-T1 rate. And while that's a T0 total, it might also be a USCMST1 total. Neither FNAL nor BNL T1 production are using SRM for access by batch jobs, but user groups might.

Gene: Who here would participate in code reviews, if such followed from this review? (No hands shot up.)

Pedro: If dCache itself is a scaling obstacle, they will deploy multiple dCaches if they have to - divided by space token or whatever.

T: The admin effort zooms! P: Ya gotta do what ya gotta do.

T&C will create some more profiling data by Wednesday, using FNAL T1.

P can do something similar; RHIC cluster has extra resources that could be used.

P: Counted 2,622 "synchronized" declarations. T: It's partly a code quality issue. G: synchronized is cheaper than it used to be in java before 1.6.

T: P suggested "memcached." Seems to be a possible way to share objects among multiple servers? S: Valentin saw a 30x improvement in speed, using memcached in front of a database.

S: Suggests a code walkthrough (as opposed to review) might be useful.

C: What are the alternatives to various outside components being used (Tomcat, Globus, ...)?

Unit tests: Hardly any exist for SRM. P volunteers to provide some, and suggests ways to use them for profiling.

G: On the subject of making the code approachable, there are already sequence diagrams. Could add a flow map that shows the components invoked by requests.

Day 2, March 18, 2009

=====

Present: All reviewers, Gene Oleynik, Timur Perelmutov

Conferencing conquered at 8:15

Mail from Gerd: In addition to memcached look at Terracotta

<http://www.terracotta.org/>

GB: Results - what can we conclude from them? TP,GB: Agree biggest user of CPU is the on authentication. But is the CPU a bottleneck? What frequency did we achieve? Didn't measure transfer rate

SG: Will the system scale by adding more servers.

GB: No evidence that system would scale with servers.

GB: What rate can you push SRM ping to until you max out CPU. How far can we push name service (how many getmetadata ops/s)

GB: Push frequency of ops up, how does the CPU load respond

TL/Timur: Use Gridworks

TL: Garbage collection an impact

SG: CPU limit, does it scale linear

SG: Assume it doesn't scale, what are alternatives?

GB: Should guarantee that once the system gets loaded its performance does not degrade from the maximum

SG: Why is SRM authenticating when job already has? CD: CMS uses dcap locally just for that reason. SRM spec requires GSI authentication.

CD: 3000 jobs submitted with 1 interactive job. Interactive job was slowed down or stuck. Each SRM communication was slowed down. Turn off jobs and it recovered.

TP: Context switching does not allow 100% utilization of CPU.

GB: 90% of the CPU usage is GSI related

TP: CERN is already using DNS load balancing. Apache with tomcat as back end for single installation.

GB: NGDF gets 5 Hz on a 4 core machine with 20-25% load on SRM server.

GB: Tomcat 6 async servlets. Is in supported in Axis?

GB: Call tree: how long did profile run? : Long enough that startup times not significant

PS: BNL can help with profiling.

CD: Can switch production to SRM for 5 minutes without profiler

PS: 4 CPUs, 50% if CPU with max load

PS: Main concern scale by more machines, separation of activities. Redundancy of service important to BNL as well.

SG: Note CMS has fallen back on non-standard dcap for local access due to SRM performance. Would like to have used SRM uniformly.

PS: All farm nodes at BNL use dcap. SRM for Atlas wide community.

TL: Why not use own load balancing - GSI authentication.

PS: memcached - not a reliable service (app must be made to work if it is unavailable). OK for database where there is a back end, but not for

SRM messages. GB: Terracotta can sync.

GB: Of course the load balancing will work given that the database will not be a bottleneck.

GB: More detailed design proposal? Can make a more detailed design if it is decided high-level design is thought the right thing to do.

TP: Networking can provide hardware. GB: A prototype in the framework of a month and some real transaction testing.

10 minute break

Conclusions:

gerd: What do we need to scale up to 100 Hz. Improve GSI authentication - multi-core, multiple front ends, improve code

About throughput: not convinced SRM is the bottleneck . Investigate Terracotta

Proper tools to profile what is going on, identify bottlenecks is very important, especially for when things go online.

Not opposed to distributed SRM front-end but believe it is not the only thing that will be needed.

Tanya: Agree with Gerd. Volunteers to work with Pedro and Catalin to develop monitor and tests of performance. We should talk with CERN and others to start discussions of replacing GSI authentication with SSL. Should include caching mechanism like Terracotta and included in documentation.

General SRM requests: Exception handling not very good as logged - site admins having difficult time analyzing. Same with messages SRM returns back to client. e.g. timeout when there is a missing host cert. All should look at SRM client and exit codes (fails with 0 exit code so can't script it but must parse output).

Documentation for tuning parameters and recommendations in detail for specific circumstances. Define what each thread pool is doing and the effect of its allocation.

Suzanne: Not convinced it will scale to 100Hz even with all proposed improvements. Root cause is tied to standard that will not do 100 Hz. Clarify the 100Hz requirement - what does it mean? Recommend going ahead with improvements. Look at root cause, compare to CERN. Increase staff - a person with dedicated time allocated for performance analysis. Agree with above recommendations. Exponential backoff on

transient errors from FTS and other clients.

Catalin:

Load balancer important for their production.

Customized installations based on the size of the system (by installation software). Later dynamic adjustment of parameters based on load. Requires some analysis. E.g. if pnfs is loaded scale back puts, etc. Include our knowledge learned from operation into the system.

decrease dependence of SRM on other components (reduce number of queries by caching, etc).

Pedro:

Priority: multiple front ends

Check for alternatives to GSI authentication implementation SSL instead of GSI. Talk to others

Tanya will try to organize performance measurements/monitoring

Possibility of separating activities

Can jobID put in all messages. - GB: available in 1.9.1+

Error messages: should not change text for backwards compatibility! Only change new.

GB: need a stable means of getting information

Would like an RPM with only SRM and decouple with releases

Timur should consider Terracotta proposed by Gerd.

Would like unit tests for basic functionality of SRM to be developed. TL: S2 tests by flavia. PS: Should have UT (regression) in repository with code and run to validate releases. A practical necessity for re-factoring. Pedro would like his team to participate on unit and certification tests (pending M Ernst agreement).

CERN has approached dCache about SSL authentication as an alternative. Requires we start supporting it in server.

PS: Partition of service up by activities for QOS. Performance isolation and QOS based on activities. Requires changes in scheduler.

Code review after Terracotta. Perform by small pieces: Request handler, job persistency, scheduler, authentication, database performance.

Make a pass through the code to make it more readable in a standard format, code structure. Parts of the code seem like copy paste (other than generated code), naming conventions, javadoc documentation on all, UML diagrams. Pedro would participate.

Terracotta decision should be made as soon as possible because it will affect the design.

TL: Effort.

GB: Terracotta might be applicable to other components of dCache so would

PS : Plan should be established with timelines and plan

One outcome should be a plan. And it should be followed up (by Gene or Sasha Moibenko here)

Request for a more detailed design documentation.

Gene will write this review up and send it around next week. We will accept changes during April, but will start implementing the plan starting with a decision on Terracotta (dCache too), a more detailed load balanced design, and a plan with effort and milestones.

DRAFT